Zoea

Research Note 20

Prolonging the Shelf Life of Software

Edward McDaid & Sarah McDaid
21 Mar 2022

Even the best software begins to degrade almost as soon as it is written. Why is this and what if anything can we do about it?

Software is a moving target. The requirements that define the functionality of a piece of software are rarely fixed in stone. Instead they evolve over time to reflect changes in the context within which the software operates. So unless code is updated it will increasingly fail to satisfy the prevailing requirements. We can't avoid the fact that requirements change but there are other factors that also cause software to degrade over time.

Code that is modified over a long period has a natural tendency to degenerate into a mess. For one thing it becomes inconsistent as successive modifications are made by people with different ideas about how things should be done. In addition, codebases often exhibit the cumulative impact of every corner cut to get something delivered or fixed in a hurry. This is called technical debt. Addressing technical debt delivers little immediate value - it doesn't add new features or fix any bugs so it is rarely prioritised. As a result technical debt accumulates while development becomes slower and more expensive.

Technology churn is another problem. Any system that is developed or maintained over a number of years can be impacted by changes in the availability of technology or changing fashions in how technology is applied. It can often seem like a good idea to replace a technology if an alternative offers some benefits. However, the number of technologies used and the rate of change needs to be controlled otherwise bad things can happen. Systems can end up with a plethora of components that often overlap in terms of functionality. In extreme cases the delivery of business value can grind to a halt.

Given the choice most developers have a strong preference for writing new code rather than modifying existing software. Unless it's fresh in their mind this is often the case even if they actually wrote the code in the first place. This is partly because it takes a lot of effort to reverse engineer the original requirements and understand how they have been delivered. In addition existing code can seriously constrain how any new stuff can be built. Aversion to working with existing code leads to the unsustainable alternatives of either rewriting or duplicating the current functionality.

These types of problems are widespread and they have existed as long as we have been writing code. This is because they are actually inherent in the way that we develop software. Until recently virtually all code has been written and modified by people. In an ideal world code would simply be generated directly from the requirements. An AI called Zoea now makes this vision a reality.

Test driven development is a highly regarded approach to coding in which developers produce a set of test cases before writing any software. The idea is that as development proceeds an increasing number of tests will pass until the code is complete. With Zoea developers produce test cases and that's it. Zoea uses AI to automatically generate the required code directly from the test cases.

With conventional software there are many reasons why the code changes on a faster cycle than the requirements. As a result, even though requirements do change the code is much more volatile. In Zoea the test cases are the software requirements. Any time the requirements change the corresponding software is simply regenerated. No more technical debt. No more code rot.

Learn more at **zoea.co.uk**